The Jogger's Problem: Accounting for Body Dynamics in Real-Time Motion Planning*

Andrei M. Shkel and Vladimir J. Lumelsky
University of Wisconsin-Madison
Madison, Wisconsin 53706, USA

Abstract—The existing approaches to sensor-based motion planning tend to deal solely with kinematic and geometric issues, and ignore the system dynamics. This work attempts to incorporate body dynamics into the paradigm of sensor-based motion planning. We consider the case of a point mass mobile robot operating in a planar environment with unknown stationary obstacles of arbitrary shape. Given the constraints on the robot's dynamics, sensing, and control means, conditions are formulated for generating collision-free trajectories with guaranteed convergence. The approach calls for continuous computation and is fast enough for real time implementation. Based on its velocity and sensing data, the robot continuously plans its motion based on the *canonical solution* which presents the time-optimal path within the robot's current sensing range. For a special case of a sudden potential collision an option of a safe emergency stopping path is always maintained. Simulated examples demonstrate the algorithm's performance.

I. Introduction

This work studies the effects of body dynamics on robot sensor-based motion planning, with the goal of designing provably correct algorithms for motion planning in an uncertain environment. We consider a mobile point robot operating in two-dimensional work space, $\mathcal{W} \subset \Re^2$, possibly filled with a (locally finite) number of unknown static obstacles $\mathcal{O} \in \mathcal{W}$. Each obstacle $\mathcal{O}_k$ is a simple closed curve of arbitrary shape, finite length, and such that a straight line will cross it in only a finite number of points. Obstacles do not touch each other (if they do, they are considered one obstacle).

The robot is a *point mass*, of mass m. It has sensors (say, vision or range finders) which allow it, at its current location C_i, to detect obstacles, and the distance to them, within its *sensing range* – a disc $D(C_i, r_v)$ of radius r_v ("radius of vision") centered at C_i. The task is to move in $\mathcal{W}$ from point S (start) to point T (target) (see Figure 1). At moment t_i, the robot's input information includes its current velocity vector $\mathbf{V}_i$, coordinates of C_i and T.

*This work is supported in part by the US Sea Grant R/NI-20 and DOE (Sandia Labs) Grant 18-4379C.

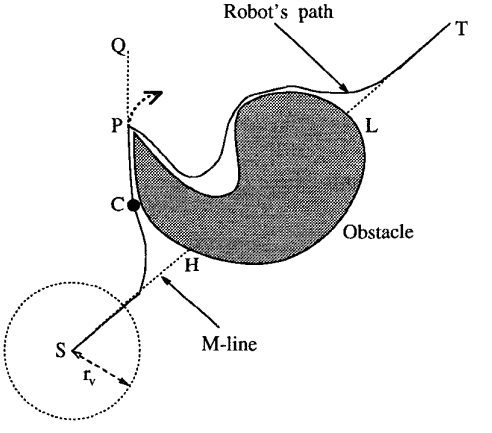

Fig. 1. Example of performance of a kinematic algorithm [1].

Motion planning is done in small *steps* $i = 0, 1, 2, ...$, (say, 30 or 50 times per second), resulting in continuous motion. Each step i takes time $\delta t = t_{i+1} - t_i = const$; the path length within time interval δt depends on the robot's velocity $\mathbf{V}_i$. Steps i and $i+1$ start at times t_i and t_{i+1}, respectively; $C_0 = S$. The motion control means are two components of the acceleration vector $\mathbf{u} = \frac{f}{m} = (p, q)$, where f the force applied. Controls $\mathbf{u}$ are constant within the step and come from a set U of measurable, piecewise continuous bounded functions in $\Re^2$, $U = \{\mathbf{u}(\cdot) = (p(\cdot), q(\cdot))/p \in [-p_{max}, p_{max}],\ q \in [-q_{max}, q_{max}]\}$. Component p controls forward (backward when braking) motion; its positive direction coincides with the robot's velocity vector $\mathbf{V}$. Component q, the steering control, is perpendicular to p forming a right pair of vectors, Figure 2.

Besides the usual issues of step-by-step planning and convergence, an additional component of planning appears because of the robot's dynamics. A move reasonable from the standpoint of the global path – for example, a sharp turn – may not be physically realizable because of the robot's inertia. The control and planning issues that appear in this model seem quite similar to those faced by

0-8186-7108-4/95 $4.00

a jogger in the city environment – hence the name *The Jogger's Problem.*

When the jogger starts turning a corner and suddenly sees a heavy large object right on the intended path, some quick replanning will take place, almost simultaneous with further sensing and motion execution. His velocity may temporarily decrease and the path will smoothly divert from the object. Or, he may "brake" to a halt, and start a detour path. Unless a right relationship is maintained between the velocity at the time of noticing the object, the distance to it, and the runner's mass, collision may occur. For a bigger mass, for example, better (further) sensing is needed to maintain the same velocity. Also, if obstacles can appear at any time and distance, and if higher velocities are essential, the control algorithm must provide an "insurance" option of a safe stopping at all times.

Most approaches to automatic motion planning adhere to one of two paradigms differing in their assumptions about the input information available. In the first paradigm, called *motion planning with complete information* (or the *Piano Mover's problem*), one assumes perfect information about the robot and obstacles, and algebraic representation of objects; motion planning is a one-time off-line operation (see, e.g., [2, 3]). Dynamics and control constraints can be incorporated into this model as well [4, 5], for example by introducing a two-stage planning process and time-optimal trajectories [6, 7].

This paper is concerned with the second paradigm, called *motion planning with incomplete information*, or *sensor-based motion planning.* Here, objects in the environment can be of arbitrary shape, and the (sensory) input information is typically of local character [1]. This paradigm naturally fits the methodology of control theory due to its notion of sensor feedback.

Techniques that ignore system dynamics can be called *kinematic*, as opposed to the *dynamic* techniques which do take dynamics into account. The existing kinematic techniques can be divided into two groups – those for *holonomic* systems and for *nonholonomic* systems [8]. A number of kinematic strategies for holonomic systems originate in maze-searching algorithms [1, 9]; when applicable, they are usually fast, can be used for real time control, and guarantee convergence; the obstacles can be of arbitrary shape. Below we make use of such algorithms.

To design a provably-correct dynamic algorithm for sensor-based motion planning, one needs a single control mechanism – separating it into stages is likely to destroy convergence. Our algorithm will operate as a single step-by-step procedure which (a) places each step on a globally converging collision-free path, while (b) satisfying the robot dynamics constraints, as follows. At the moment (step) i, the kinematic algorithm identifies an *intermediate target* point, T_i, which lies on a convergent path and, for local path optimization, is far enough from the robot – normally at the boundary of the sensing range. Then, a step is attempted toward T_i, and the process repeats. If due to inertia and occluding obstacles the current T_i goes out of the robot's sight, the robot will use *temporary intermediate targets* until it can see point T_i again. Because of dynamics, the step toward T_i may not be possible and is thus modified. Once a step is physically executed, new sensing information appears and the process repeats.

In principle, any maze-searching strategy can be utilized for the kinematic part of the algorithm. For the sake of specificity, below we make use of the VisBug algorithm [1]. Roughly, VisBug operates as follows (see Fig. 1): (1) Walk from S toward T along a straight line (called M-line) until detect an obstacle crossing the M-line, say at point H. (2) Using sensors, define the farthest visible intermediate target T_i on the obstacle boundary; make a step toward T_i; iterate Step 2 until detect M-line; go to Step 1. In Fig. 1, note that while trying to pass the obstacle from the left, at point P the robot will make a sharp turn.

The step planning task is formulated as an optimality problem. At each step, a *canonical solution* is found which, with no obstacles present, would bring the robot from C_i to T_i with zero velocity and in minimum time. If the canonical path crosses an obstacle and is thus not feasible, a *near-canonical solution* path is found which is collision-free and satisfies the control constraints. We show, first, that in this case only a finite number of option paths needs be considered, and second, there exists at least one path solution that is collision-free. By decoupling the two-dimensional control problem into two one-dimensional control problems, an extremely fast procedure, easily implementable in real time, is produced for finding the multi-step time-optimal path within the sensing range.

We define three coordinate systems (follow Figure 2):

– The *world frame*, $(\mathbf{x}, \mathbf{y})$, fixed at point S.

– The *primary path frame*, $(\mathbf{t}, \mathbf{n})$, is a moving (inertial) coordinate frame. Its origin is attached to the robot; axis $\mathbf{t}$ is aligned with the current velocity vector $\mathbf{V}$, axis $\mathbf{n}$ is normal to $\mathbf{t}$. Together with axis b, which is a cross product $\mathbf{b} = \mathbf{t} \times \mathbf{n}$, the triple $(\mathbf{t}, \mathbf{n}, \mathbf{b})$ forms the *Frenet trihedron*, with the plane of $\mathbf{t}$ and $\mathbf{n}$ forming the *osculating plane* [10].

– The *secondary path frame*, (ξ_i, η_i), is a coordinate frame that is fixed during the time interval of the step i.The frame's origin is at the intermediate target T_i, with axis ξ_i aligned with the velocity vector $\mathbf{V}_i$ at time t_i, and axis η_i normal to ξ_i.

For convenience, we combine the requirements and constraints that affect the control strategy into a set, called Ω. A solution (a path, a step, or a set of control values) is said to be Ω-*acceptable* if, given the current position C_i and velocity $\mathbf{V}_i$, (i) it satisfies the constraints on controls,

$|p| \le p_{max}$, $|q| \le q_{max}$; (ii) it guarantees a stopping path; (iii) it results in a collision-free motion.

Safety considerations due to dynamics appear in a number of ways. Since no information about the obstacles is available beyond distance r_v from the robot, guaranteeing collision-free motion means assuring at any moment at least one "last resort" *stopping path* that would bring the robot to a halt within the radius r_v if needed. If this rule is violated, at the next step new obstacles may appear in the sensing range, such that collision will become imminent no matter what control is used. This dictates a certain relation between the velocity $\mathbf{V}$, mass m, and controls $\mathbf{u} = (p, q)$: if the robot moves with the maximum velocity, the stop point of the stopping path must be no further that at distance r_v from the current position C. This relation is $V_{max} = \sqrt{2 p_{max} r_v}$.

In Fig. 1, when approaching point P, the robot will designate it as its next intermediate target T_i. For awhile T_i will stay at P because no other visible point on the obstacle boundary appears until the robot arrives at P. During this time, unless a stopping path is possible at any time, the robot would have to plan to stop at P. Otherwise, it may arrive at P with a non-zero velocity, start turning around the corner, and suddenly uncover an obstacle invisible so far, making a collision unavoidable.

Convergence. Because of dynamics, the convergence mechanism borrowed from a kinematic algorithm needs some modification. For example, the robot's inertia may cause it to move so that the intermediate target T_i will become invisible, either because it goes outside the sensing range r_v (as after point P, Fig. 1), or due to occluding obstacles (Fig. 5), and so the robot may lose it (and the path convergence with it). The solution chosen is to keep the velocity high and, if the intermediate target T_i does go out of sight, modify the motion locally until T_i is found again. [1]

II. Dynamics and collision avoidance.

Consider a time sequence $\sigma_t = \{t_0, t_1, t_2, ...\}$. Step i corresponds to the interval $[t_i, t_{i+1})$. At moment t_i the robot is at the position C_i, with the velocity vector $\mathbf{V}_i$. Within this interval, based on the sensing data, the intermediate target T_i (supplied by the VisBug procedure), and vector $\mathbf{V}_i$, the control system calculates the values of controls p and q, applies them to the robot, and the robot executes step i, finishing it at point C_{i+1} at moment t_{i+1}, with the velocity vector $\mathbf{V}_{i+1}$; then the process repeats.

Below, we first develop the equations of motion for a step interval i. Then the *canonical solution* is obtained, and finally, the *near-canonical solution*, for the case when obstacles interfere with the canonical solution.

[1] Some details and proofs, omitted due to space limitations, can be found in [11].

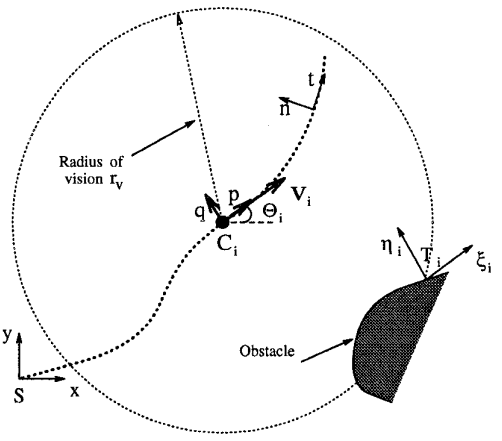

Fig. 2. Coordinate frames: $(\mathbf{x}, \mathbf{y})$ is the world frame, with its origin at S; $(\mathbf{t}, \mathbf{n})$ is the primary path frame; for the current position C_i, (ξ_i, η_i) is the secondary path frame.

The equations of motion. The remainder of this section refers to the time interval $[t_i, t_{i+1})$, and so the index i can be dropped. Denote $(x, y) \in \mathbf{R}^2$ the position of the robot in the world frame, and θ the (slope) angle between the velocity vector $\mathbf{V} = (V_x, V_y) = (\dot{x}, \dot{y})$ and x-axis of the world frame (see Fig. 2). The planning process involves computation of the controls $\mathbf{u} = (p, q)$, which for every step defines the velocity vector and eventually the path, $\mathbf{x} = (x, y)$, as a function of time. The angle θ between vector $\mathbf{V} = (V_x, V_y)$ and x-axis of the world frame is found as

$$\theta = \begin{cases} \arctan(\frac{V_y}{V_x}), & V_x \ge 0 \\ \arctan(\frac{V_y}{V_x}) + \pi, & V_x < 0 \end{cases}$$

Given that the control components p and q act along the $\mathbf{t}$ and $\mathbf{n}$ directions, respectively, the equations of motion with respect to the path frame are $\dot{V} = p, \quad \dot{\theta} = q/V$. Since vector (p, q) is constant over time interval $[t_i, t_{i+1})$, within this interval the solution for $V(t)$ and $\theta(t)$ becomes

$$\begin{aligned} V(t) &= pt + V^0, \\ \theta(t) &= \theta^0 + \frac{q \log(1 + \frac{tp}{V})}{p} \end{aligned} \qquad (1)$$

where θ^0 and V^0 are constants of integration and are equal to the values of $\theta(t_i)$ and $V(t_i)$, respectively. By parameterizing the path by the value and direction of the velocity vector, the path can be mapped onto the world frame (x, y) using the following vector integral equation:

$$\mathbf{r}(t) = \int_{t_i}^{t_{i+1}} V \mathbf{t} dt \qquad (2)$$

Here $\mathbf{r}(t) = (x(t), y(t))$, and $\mathbf{t} = (\cos(\theta), \sin(\theta))$ is a projection of unit vector along the $\mathbf{V}$ direction. After integrating equation (2), we obtain the set of solutions of the

form:

$$x(t) = \frac{2p\cos\theta(t) + q\sin\theta(t)}{4p^2+q^2}V^2(t) + A$$
$$y(t) = -\frac{q\cos\theta(t) - 2p\sin\theta(t)}{4p^2+q^2}V^2(t) + B \qquad (3)$$

where terms A and B are

$$A = x_0 - \frac{{V_0}^2\,(2\,p\,\cos(\theta_0) + q\,\sin(\theta_0))}{4\,p^2+q^2},$$
$$B = y_0 + \frac{{V_0}^2\,(q\,\cos(\theta_0) - 2\,p\,\sin(\theta_0))}{4\,p^2+q^2}$$

$V(t)$ and $\theta(t)$ in equations (3), which are directly controlled by the values of p and q, are given by equation (1).

In the general case, equations (3) describe a spiral curve. Note two special cases: when $q = 0, p \neq 0$, equations (3) describe a straight line motion along the vector of velocity; when $p = 0, q \neq 0$, they produce a circle of radius $\frac{V_0^2}{|q|}$ centered at the point (A, B).

Canonical solution. This solution presents a path which, assuming no obstacles, will bring the robot from C_i to T_i with zero velocity and in minimum time. The assumption of L_∞-norm allows us to decouple the bounds on accelerations in ξ and η directions, and thus treat the two-dimensional problem as a set of two one-dimensional problems.

The optimization problem is formulated based on the Pontryagin's optimality principle, with respect to the secondary frame (ξ, η). We seek to optimize the criterion F which signifies time. Assume the trajectory being sought starts at time $t = 0$, at point $(\xi_0, \eta_0, \dot\xi_0, \dot\eta_0)$, and ends at time $t = t_f$ (f for "final"), at the phase-space origin. Then, the problem in hand is described as follows:

$$F(\xi(\cdot), \eta(\cdot), t_f) = t_f \to \inf,$$
$$\ddot\xi = p, \quad \|p\| \le p_{max},$$
$$\ddot\eta = q, \quad \|q\| \le q_{max},$$
$$\xi(0) = \xi_0, \quad \eta(0) = \eta_0, \quad \dot\xi(0) = \dot\xi_0, \quad \dot\eta(0) = \dot\eta_0,$$
$$\eta(t_f) = \eta(t_f) = \dot\xi(t_f) = \dot\eta(t_f) = 0$$

Analysis shows that the optimal solution of each one-dimensional problem corresponds to the "bang-bang" control, with at most two switches along the ξ and η directions, at times $t_{s,\xi}$ and $t_{s,\eta}$ ("s" for "switch"), respectively.

The switch curves for control switchings are two connected parabolas:

$$\mu = -\frac{\dot\mu^2}{2g_{max}}, \quad \dot\mu > 0$$
$$\mu = \frac{\dot\mu^2}{2g_{max}}, \quad \dot\mu < 0 \qquad (4)$$

where μ and g_{max} are to be replaced by ξ and p_{max}, respectively, in case of $(\xi, \dot\xi)$ space, and by η and q_{max}, in case of $(\eta, \dot\eta)$ space.

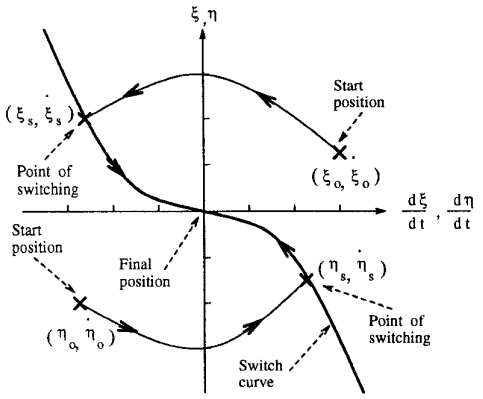

Fig. 3. Examples of start positions for both phase spaces, $(\xi, \dot\xi)$ and $(\eta, \dot\eta)$. For $(\xi, \dot\xi)$, the start position is above the switch curves; for $(\eta, \dot\eta)$ it is under the switch curves.

The time optimal solution is then obtained using the bang-bang strategy for ξ and for η, respectively, depending on whether the starting points, $(\xi_o, \dot\xi_o)$ and $(\eta_o, \dot\eta_o)$ are above or below their corresponding switch curves, as follows:

$$\hat p(t) = \begin{cases} \alpha_1 \cdot p_{max}, & 0 \le t \le t_{s,\xi} \\ \alpha_2 \cdot p_{max}, & t_{s,\xi} < t \le t_f \end{cases}$$
$$\hat q(t) = \begin{cases} \beta_1 \cdot q_{max}, & 0 \le t \le t_{s,\eta} \\ \beta_2 \cdot q_{max}, & t_{s,\eta} < t \le t_f \end{cases} \qquad (5)$$

where $\alpha_1 = -1, \alpha_2 = 1$ if $(\xi_o, \dot\xi_o)$ is above (respectively, $\alpha_1 = 1, \alpha_2 = -1$ if $(\xi_o, \dot\xi_o)$ is below) its switch curve, and $\beta_1 = -1, \beta_2 = 1$ if $(\eta_o, \dot\eta_o)$ is above ($\beta_1 = 1, \beta_2 = -1$ if $(\eta_o, \dot\eta_o)$ is below) its switch curve. For example, if the initial conditions for ξ and η are as shown in Figure 3, then $\alpha_1 = -1, \alpha_2 = 1$ and $\beta_1 = 1, \beta_2 = -1$.

The time, position and velocity of the control switching for the ξ components are described by

$$t_{s,\xi} = \frac{\sqrt{\frac{(\dot\xi_0)^2}{2} + \xi_0 p_{max}} + \dot\xi_0}{p_{\max}}$$
$$(\xi_s, \dot\xi_s) = \left(\frac{\dot\xi_0^2}{4p_{max}} + \frac{\xi_0}{2}, -\sqrt{\frac{\dot\xi_0^2}{2p_{max}} + \xi_0 p_{max}}\right)$$

and those for the η components by

$$t_{s,\eta} = \frac{\sqrt{\frac{(\dot\eta_0)^2}{2} - \eta_0 q_{max}} - \dot\eta_0}{q_{\max}}$$
$$(\eta_s, \dot\eta_s) = \left(-\frac{\dot\eta_0^2}{4q_{max}} + \frac{\eta_0}{2}, \sqrt{\frac{\dot\eta_0^2}{2q_{max}} - \eta_0 q_{max}}\right)$$

The number, time, and locations of switchings can be uniquely defined from the initial and final conditions. It

can be shown [11] that for every position of the robot in the phase space $\Re^4$ – space of position and velocity, $(\xi, \eta, \dot{\xi}, \dot{\eta})$ – the obtained control law will guarantee time-optimal motion in both ξ and η directions, as long as the time interval considered is sufficiently small. Substituting this control law in the equations of motion (3) produces the canonical solution.

In summary, the procedure is as follows: (a) Substitute the current position/velocity $(\xi, \eta, \dot{\xi}, \dot{\eta})$ for μ in (4); see if the starting point is above or below the switch curves. (b) Depending on (a), take one of the four bang-bang control pairs (p, q), as in (5). (c) With this pair (p, q), find from (3) the position C_{i+1} and from (1) the velocity $\mathbf{V}_{i+1}$ and angle θ_{i+1} at the end of the step. If this step to C_{i+1} crosses no obstacles, and if there exists a stopping path in the direction $\mathbf{V}_{i+1}$, the step is accepted; otherwise, a near-canonical solution is sought, see below.

Note that though the canonical solution defines a fairly complex multi-step path from C_i to T_i, only one – the very first – step of that path is calculated explicitly. The switch curves (4) and the position and velocity equations (1), (3) are quite simple. The whole computation is remarkably fast.

Near-canonical solution. If the stopping path of the candidate step happens to cross an obstacle within the distance $d < V_i^2/2p_{max}$, the controls are modified into a *near-canonical solution* that is both Ω-acceptable and reasonably close to the canonical solution. The near-canonical solution is one of the nine possible combinations of the bang-bang control pairs $(k_1 \cdot p_{max}, k_2 \cdot q_{max})$, where k_1, k_2 come from the set $\{-1, 0, 1\}$ (see Figure 4). This set is guaranteed to contain an Ω-acceptable solution: since the current position has been chosen so as to guarantee a stopping path, such a solution – for example, with $(-p_{max}, 0)$ – always exists. Further, the position of the intermediate target T_i relative to vector $\mathbf{V}_i$ – in its left or the right semi-plane – suggests an ordered and thus shorter search among the control pairs.

III. The Algorithm

The algorithm includes three procedures: the *Main body* analyzes the path towards the intermediate target T_i; *Define Next Step* chooses the components p and q; *Find Lost Target* deals with the case when T_i goes out of the robot's sight. Also used is a procedure called *Compute* T_i, from the VisBug algorithm [1], for computing the next intermediate target T_{i+1} and analyzing the target reachability. Vector $\mathbf{V}_i$ is the current vector of velocity, T is the robot's target.

Procedure *Main Body:* At each step i:

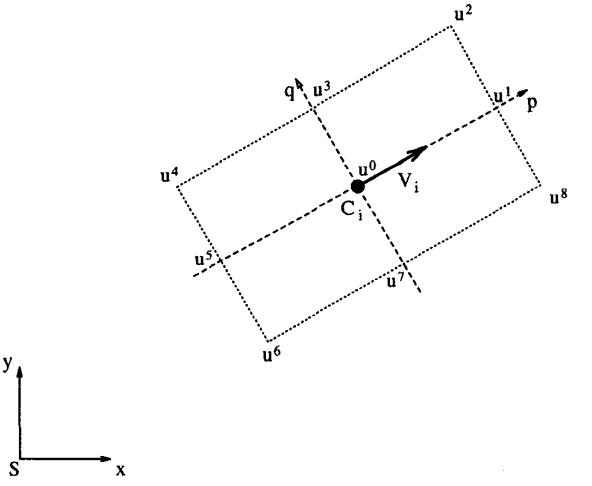

Fig. 4. Near-canonical solution. Controls (p, q) are assumed to be L_∞-norm bounded on the small interval of time. The choice of (p, q) is among eight "bang-bang" solutions shown.

If $C_i = T$, stop.
Find T_i from *Compute* T_i.
If T is found unreachable, stop.
If T_i is visible, find C_{i+1} from *Define Next Step*; make a step towards C_{i+1}; iterate; else,
Use *Find Lost Target* to make T_i visible; iterate.

Procedure *Define Next Step*: At step i:

- Step 1: Find the canonical solution (the switch curves and controls (p, q)) using equations (4), (5). If it is Ω-acceptable, exit; else go to S2.
- Step 2: Find the near-canonical solution as described above; exit.

Find Lost Target is executed when T_i becomes invisible. The last position C_i where T_i was visible is stored until T_i becomes visible again. After losing T_i, the robot keeps moving ahead while defining temporary intermediate targets on the visible part of the line segment (C_i, T_i), and continuing looking for T_i. If it finds T_i, it moves directly toward it, Fig. 5a; otherwise, if the whole segment (C_i, T_i) becomes invisible, the robot brakes to a stop and returns to C_i etc., Fig. 5b. The procedure includes these steps:

- Step 1: If segment (C_i, T_i) is visible, define on it and move toward temporary intermediate targets T_i^t, while looking for T_i. If current position $C_j = T$, exit; else if C_i lies in the segment (C_i, T_i), exit. Else go to Step 2.
- Step 2: If segment (C_i, T_i) is invisible, initiate a stopping path and then go back to C_i; exit.

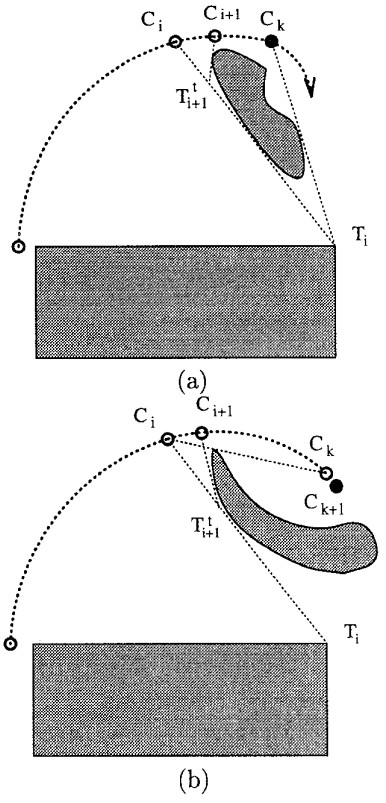

Fig. 5. In these examples, because of the system inertia the robot temporarily "loses" the intermediate target point T_i.

Convergence. The collision-free motion along the path is guaranteed by the design of the canonical and near-canonical solutions. To prove convergence, we need to show that the algorithm will find a path to the target position T if one exists, or it will infer in finite time the nonreachability of T if true. This is guaranteed by the convergence properties of the kinematic algorithm [1]. The following statements hold:

Claim 1 *Under the algorithm, assuming zero velocity at the start point, $\mathbf{V}_S = 0$, at every step of the path there exists at least one stopping path.*

Claim 2 *The algorithm guarantees convergence.*

Computational complexity. As with other on-line sensor-based algorithms, it is not very informative to assess the algorithm complexity the way it is usually done with algorithms with complete information [12]. This is because in the latter one deals with one-time computation, whereas in the former the important complexity measure is for the computations at each step – the total computation time is simply a linear function of the path length. (Recall that with the sampling rate, say, 50 per second, each step calculation must be done within the 20 msec interval.)

Though the canonical solution found at each path step by the algorithm is the solution of a fairly complex time-optimal problem, its computational cost is remarkably low, thanks to the (optimal) bang-bang control. This computation (see Section II) includes substituting the initial conditions $(\xi, \eta, \dot{\xi}, \dot{\eta})$ into the equations for parabolas (4) to see if the start point is above or below the corresponding parabola, and then simply taking the corresponding control pair $(\hat{p}, \hat{q}$ from the four choices in (5). The parabolas equations themselves are found beforehand, only once. The near-canonical solution, when needed, is similar and very fast. Note that a single step computation is of constant time: though the canonical solution represents the whole multi-step trajectory within the sensing range of radius r_v, the computation time is independent of the value r_v and the length of path within the sensing range.

IV. Examples

In the simulated examples in Figure 6, the robot's mass and constraints on the control parameters are the same for all cases. The generated paths are shown in thicker lines. For comparison, thin lines show the corresponding paths produced under the same conditions by the kinematic algorithm VisBug [1].

The difference between the examples in Figures 6a,b is that in (b) there are additional obstacles which the robot suddenly uncovers at a close distance when turning around corner; the radius of vision r_v is the same for both (a) and (b). Note that in (b) the path becomes tighter, shorter, though it takes longer: measured in the number of steps, the path in (a) takes 242 steps, and in (b) 278 steps; one might say the robot becomes more cautious in (b). A pair of examples in Figures 6c,d illustrate the effect of sensing distance: in (c) and (d) r_v is twice that of (a) and (b). The path execution times are 214 and 244 steps, respectively, shorter then in the corresponding examples (a), (b). That is, better sensing (larger r_v) resulted here in shorter time to complete the task; more crowded space resulted in longer time, though perhaps in shorter paths. Note that few times – such as at points A,B,C,D,E,F, Figure 6b – the robot found it necessary to make use of the stopping path – those points are usually easy to recognize from the sharp turns in the path (this would be impossible with a nonzero velocity).

References

[1] V. Lumelsky and T. Skewis. Incorporating range sensing in the robot navigation function. *IEEE Trans. on Systems, Man, and Cybernetics*, 20(5):1058–1069, September 1990.

[2] J. Schwartz and M. Sharir. On the "Piano Movers" problem. II. General techniques for computing topological properties of real algebraic manifolds. *Advances in Applied Mathematics*, 4:298–351, 1983.

[3] J. Canny. A new algebraic method for robot motion planning and real geometry. *Proc. 28th IEEE Symposium on Foundations of Computer Science*, 1987. Los Angeles, CA.

[4] E. Gilbert and D. Johnson. Distance functions and their applications to robot path planning in the presence of obstacles. *IEEE Journal of Robotics and Automation*, March 1985.

[5] Z. Shiller and H.H. Lu. Computation of path constrained time optimal motions along specified paths. *ASME Journal of Dynamic Systems, Measurement and Control*, 114(3):34–40, 1992.

[6] B. Donald and P. Xavier. A provably good approximation algorithm for optimal-time trajectory planning. *Proc. IEEE Intern. Conf. on Robotics and Automation*, May 1989. Scottsdale, AZ.

[7] Z. Shiller and S. Dubowsky. On computing the global time optimal motions of robotic manipulators in the presence of obstacles. *IEEE Trans. on Robotics and Automation*, 7(6):785–797, 1991.

[8] D.T. Greenwood. *"Principles of Dynamics"*. Prentice-Hall, New York, 1965.

[9] A. Sankaranarayanan and M. Vidyasagar. Path planning for moving a point object amidst unknown obstacles in a plane: A new algorithm and a general theory for algorithm development. *Proc. 29th IEEE Intern. Conf. on Decision and Control*, 1990. Honolulu, HI.

[10] G. Korn and T. Korn. *"Mathematical Handbook"*. McGraw-Hill, New York, 1968.

[11] A. Shkel and V. Lumelsky. The Jogger's Problem: Accounting for body dynamics in real-time motion planning. Technical Report RL-94007, Robotics Laboratory, University of Wisconsin-Madison, December 1994.

[12] J. Canny, A. Rege, and J. Reif. An exact algorithm for kinodynamic planning in the plane. *Proc. 6th Annual Symposium on Computational Geometry*, June 1990. Berkeley, CA.

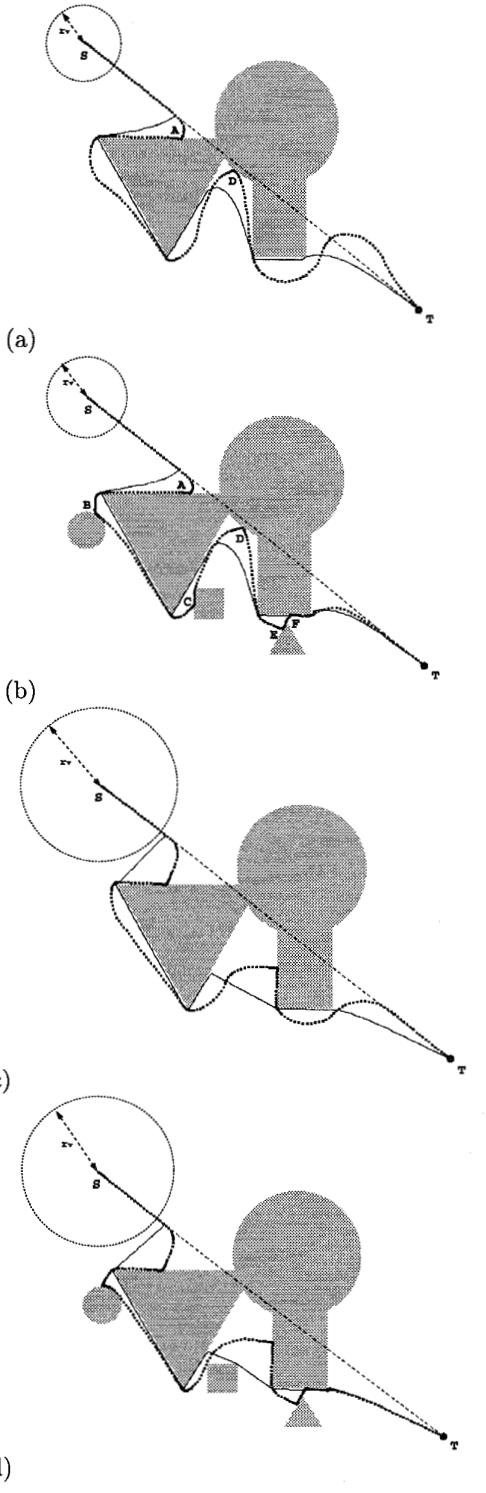

Fig. 6. Simulated examples of the algorithm's performance.